

FIG. 12

```
typedef struct EncoderObj {
    IALG_Obj ialgObj; /* IALG object MUST be first field */
    Int *workBuf; /* pointer to on-chip scratch memory */
    Int *historyBuf; /* previous frame's data in ext mem */
    ... ;
} EncoderObj ;

Void algActivate (IALG_Handle handle)
{
    EncoderObj *inst = (EncoderObj *)handle;
    /* copy history to beginning of on-chip working buf */
    memcpy(inst->workingBuf, inst->histBuf, HISTSIZE) ;
}

Void encode (IALG_Handle handle,
             Void *in[] , Void *out[] )
{
    EncoderObj *inst = (EncoderObj *) handle;

    /* append input buffer to history in on-chip workBuf */
    memcpy (inst->workBuf + HISTSIZE, in, HISTSIZE) ;

    /* encode data */
    ...
    /* move history to beginning of workbuf for next frame */
    memcpy (inst->workBuf, inst->workingBuf + FRAMESIZE, HISTSIZE) ;
}

Void algDeactivate (IALG_Handle handle)
{
    EncoderObj *inst = (EncoderObj *) handle;
    /* save beginning of on-chip workBuf to history */
    memcpy (inst->histBuf, inst->workingBuf, HISTSIZE) ;
}
```

1201

1202

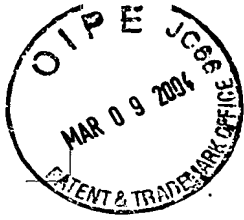


FIG. 13

```

typedef struct EncoderObj {
    IALG_Obj  ialgObj;
    Int      *workBuf;
    Int      workBufLen;
    ...;
} EncoderObj;

typedef struct EncoderParams {
    Int frameDuration;
};
EncoderParams ENCODERATTRS = {5};

Int algAlloc(IALG_Params *algParams, IALG_Fxns **p, IALG_MemRec memTab[] )
{
    EncoderParams *params = (EncoderParams *)algParams;
    if (params == NULL) {
        params = &ENCODERATTRS;
    }
    memTab [0].size = sizeof (EncoderObj) ;
    memTab [0].alignment = 1;
    memTab [0].type = IALG_PERSIST; 1301
    memTab [0].space = IALG_EXTERNAL; 1302

    memTab [1].size = params->frameDuration * 8 * sizeof(int) ;
    memTab [1].alignment = 1;
    memTab [1].type = IALG_PERSIST; 1303
    memTab [1].space = IALG_DARAM; 1304

    return (2);
}

```

/* IALG object MUST be first field */
 /* pointer to on-chip scratch memory */
 /* expressed in words per frame */

/* expressed in ms per frame */

/* default parameters */

/* use default parameters */

/* no alignment */

/* dual-access on-chip */

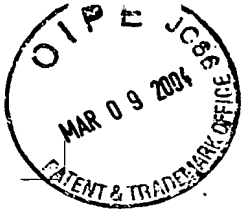


FIG. 14

```
typedef struct EncoderStatus {
    Bool voicePresent; /* voice in current frame? */
    ... ;
} EncoderStatus;

typedef enum {EncoderGetStatus, ...} EncoderCmd;

Void algControl (IALG_Handle handle,
                 IALG_Cmd cmd, IALG_Status *status)
{
    EncoderStatus *sptr = (EncoderStatus *)status;

    switch ((EncoderCmd)cmd) {
        case EncoderGetStatus:
            sptr->voicePresent = ...;
            ...
        case EncoderSetMIPS:
            ...
    }
}
```

FIG. 15

```
typedef struct EncoderObj {
    IALG_Obj   ialgObj; /* IALG object MUST be first field */
    Int        *workBuf;
    Int        workBufLen;
    ... ;
} EncoderObj;

Int algFree(IALG_Handle handle, IALG_MemRec memTab[])
{
    EncoderObj *inst = (EncoderObj *)handle;

    algAlloc(NULL, memTab); /* get default values first */

    memTab[1].size = inst->workBufLen * sizeof(Int);
    memTab[1].base = (Void *)inst->workBuf;
    return(2);
}

Int algAlloc(IALG_Params *params, IALG_MemRec memTab[])
{
    memTab[0].size = sizeof (EncoderObj);
    memTab[0].alignment = 1;
    memTab[0].type = IALG_PERSIST;
    memTab[0].space = IALG_EXTERNAL;

    memTab[1].size = 80; /* 10ms @ 8KHz */
    memTab[1].alignment = 1; /* no alignment */
    memTab[1].type = IALG_PERSIST;
    memTab[1].space = IALG_DARAM; /* dual-access on-chip */

    return (2);
}
```

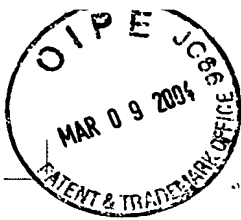


FIG. 16

```
typedef struct EncoderObj {
    IALG_Obj   ialgObj;           /* IALG object MUST be first field */
    Int        workBuf;           /* pointer to on-chip scratch memory */
    Int        workBufLen;        /* workBuf length (in words) */
    ... ;
} EncoderObj ;

Int algInit(IALG_Handle handle,
            IALG_MemRec memTab[], IALG_Handle p, IALG_Params *algParams)
{
    EncoderObj *inst = (EncoderObj *)handle;
    EncoderParams *params = (EncoderParams *)algParams;

    if (params == NULL) {
        params = &ENCODERATTRS;    /* use default parameters */
    }

    inst->workBuf = memTab[1].base;
    inst->workBufLen = params->frameDuration * 8;
    ...

    return (IALG_EOK) ;
}
```

FIG. 17

```
typedef struct EncoderObj {
    IALG_Obj   ialgObj;           /* IALG object MUST be first field */
    int        workBuf;           /* pointer to on-chip scratch memory */
    Int        workBufLen;        /* workBuf length (in words) */
    ... ;
} EncoderObj ;

1701 { algMoved(IALG_Handle handle,
            IALG_Params *algParams, IALG_MemRec memTab[] )
    {
        EncoderObj *inst = (EncoderObj *)handle;

        inst->workBuf = memTab[1].base;
    }
}
```

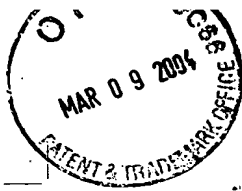


FIG. 18

```
#define NUMBUF 3          /* max number of my memory requests */
extern IALG_Fxns *subAlg; /* sub-algorithm used by this alg */

1801 {
    Int algNumAlloc(Void)
    {
        return (NUMBUF + subAlg->algNumAlloc());
    }

    Int algAlloc(const IALG_Params *p, struct IALG_Fxns **pFxns,
                 IALG_MemRec memTab)
    {
        Int n;

        /* initialize my memory requests */

        /* initialize sub-algorithm's requests */
        n = subAlg->algAlloc(..., memTab);
        return (n + NUMBUF);
    }
}
```

FIG. 19

```
/*-----*/
/*      TYPES AND CONSTANTS      */
/*-----*/
#define IRTC_ENTER          0
#define IRTC_CLASS1         1
#define IRTC_CLASS2         2
#define IRTC_CLASS3         3
#define IRTC_CLASS4         4
#define IRTC_CLASS5         5
#define IRTC_CLASS6         6
#define IRTC_CLASS7         7

/*
 * ===== IRTC_Handle =====
 * Handle to module's trace instance object
 */
typedef struct IRTC_Obj *IRTC_Handle;

/*
 * ===== IRTC_Mask =====
 */
typedef LgUns IRTC_Mask ;

/*
 * ===== IRTC_Fxns =====
 */
1901 {
    typedef struct IRTC_Fxns {
        Void      *implementationId;
        Void      (*rtcBind) (LOG_Obj *log) ;
        IRTC_Mask (*rtcGet) (IRTC_Handle) ;
        Void      (*rtcSet) (IRTC_Handle, IRTC_Mask mask) ;
    } IRTC_Fxns;
}
```

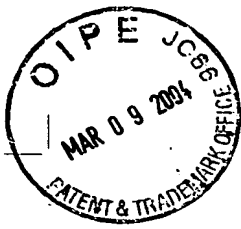
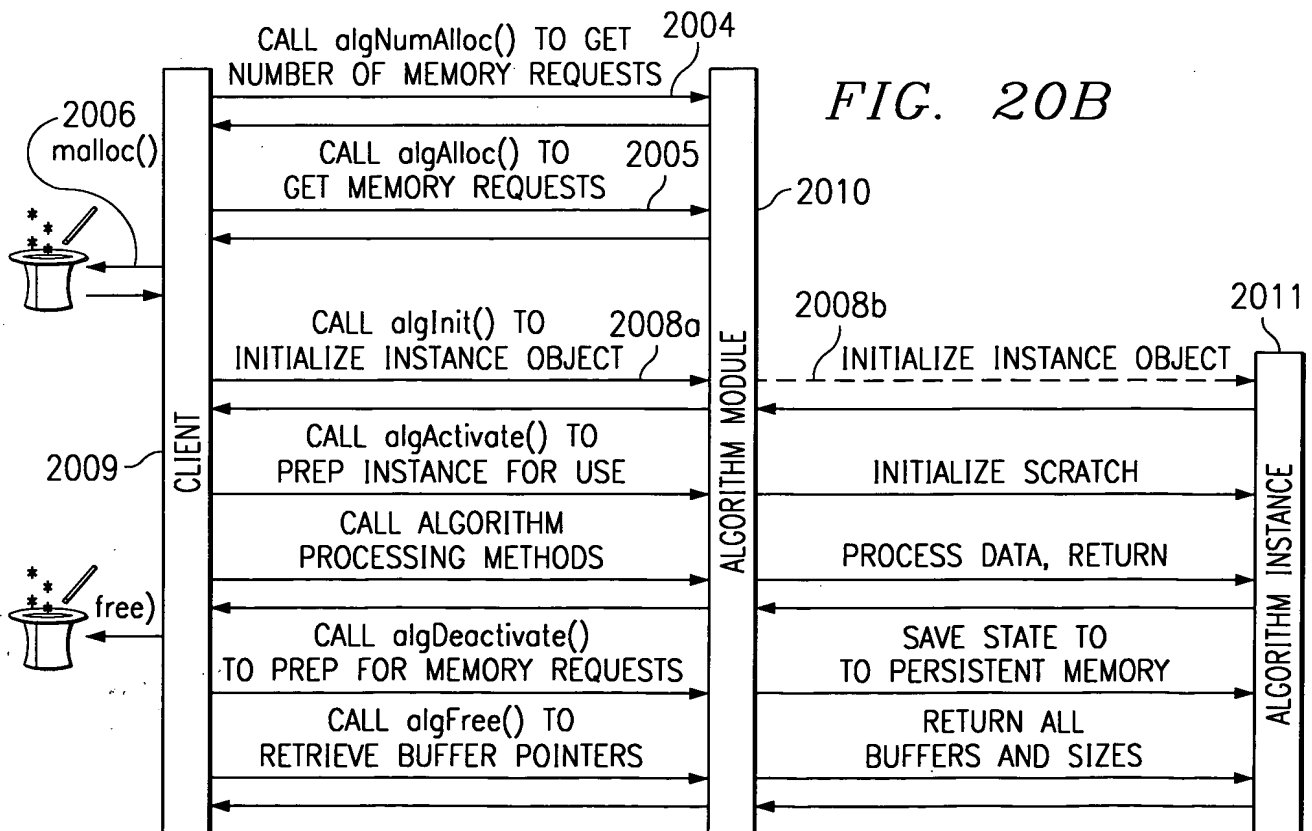
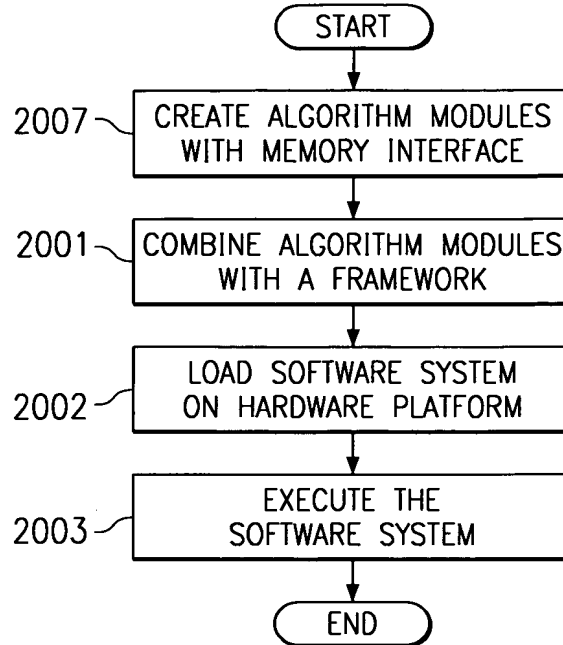


FIG. 20A



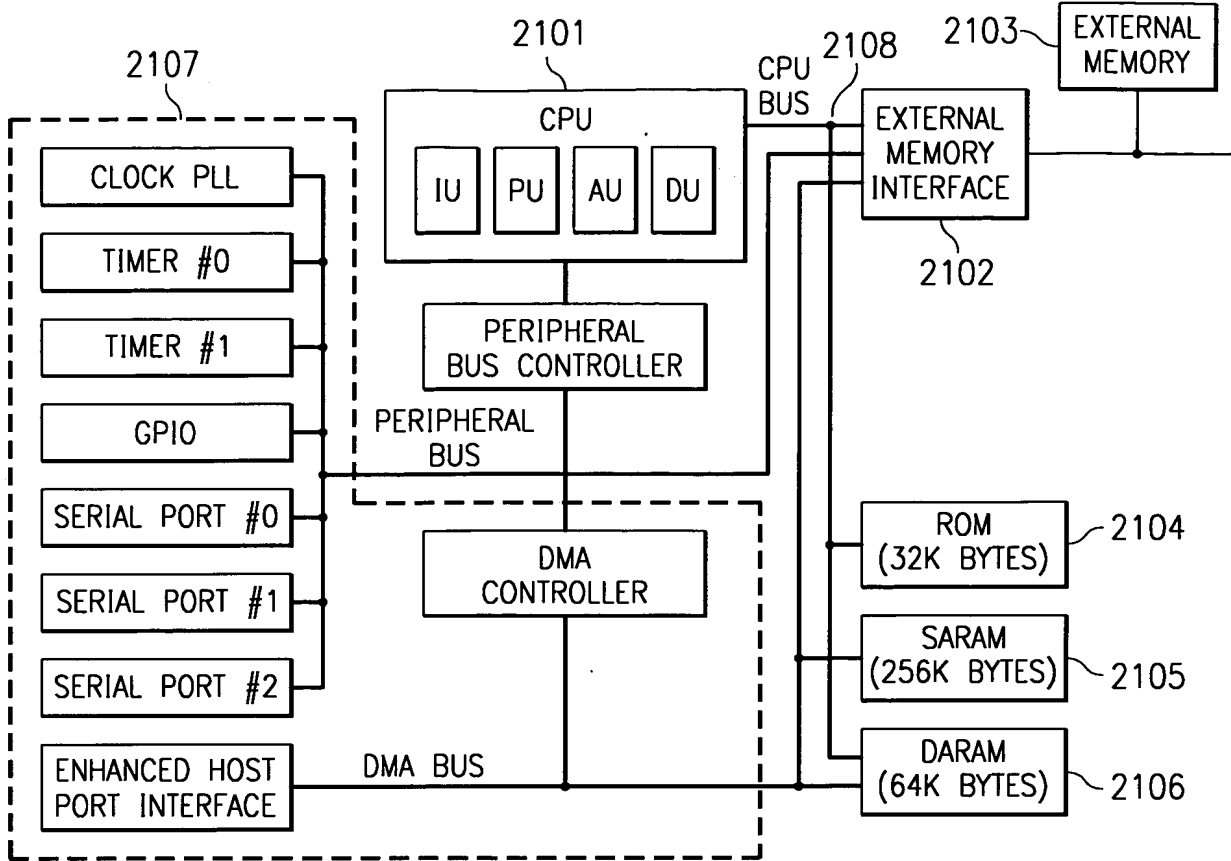


FIG. 21

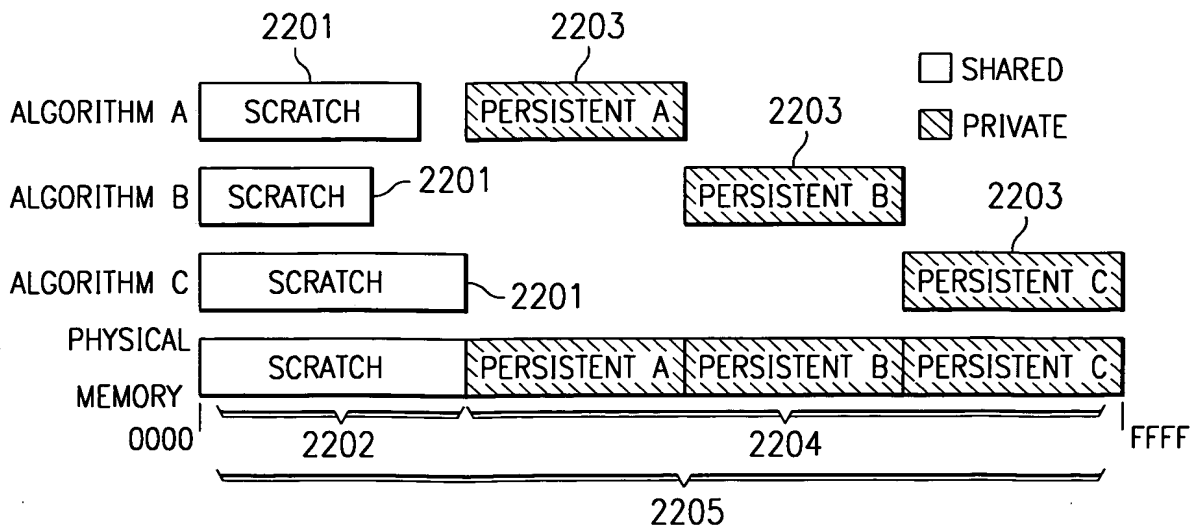


FIG. 22



FIG. 23A

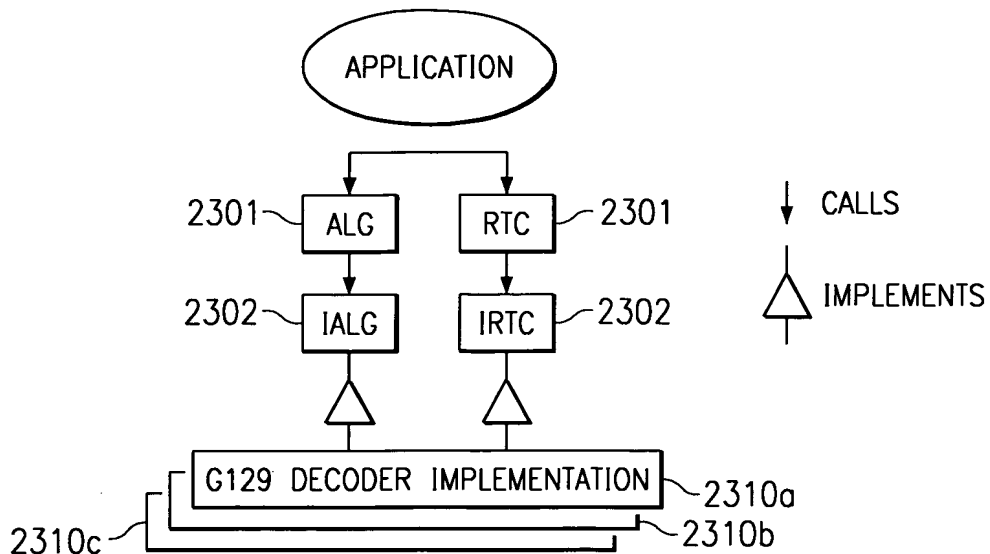


FIG. 23B

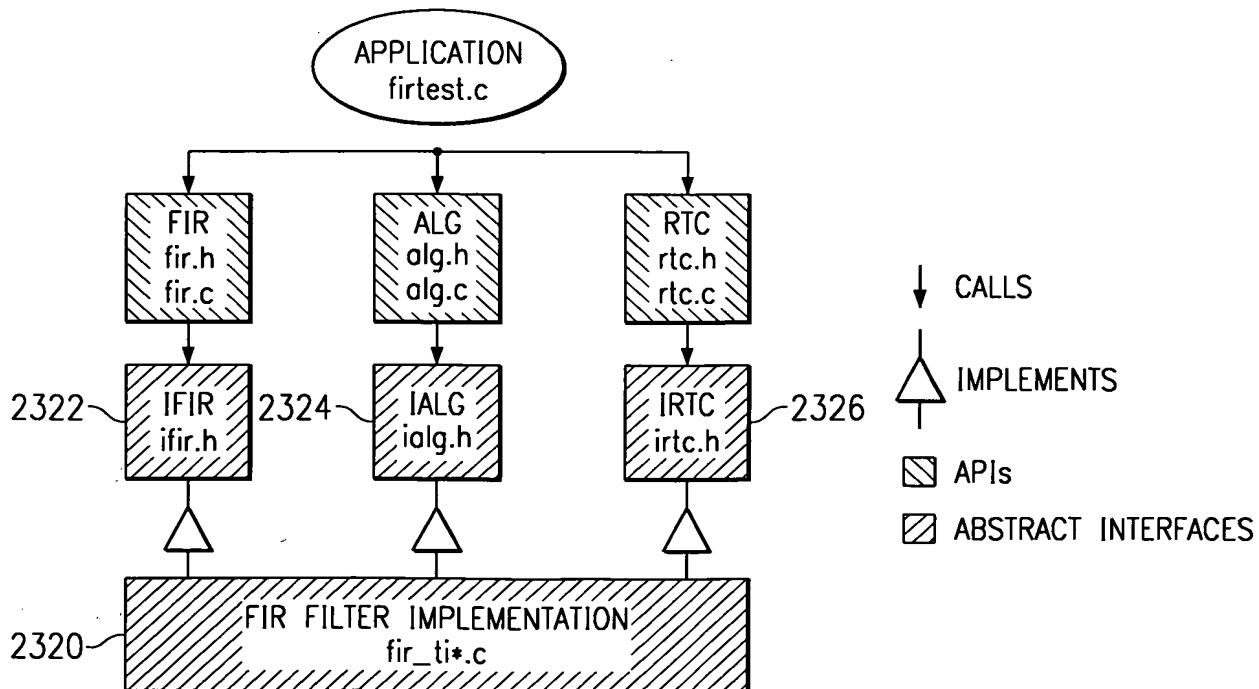




FIG. 24

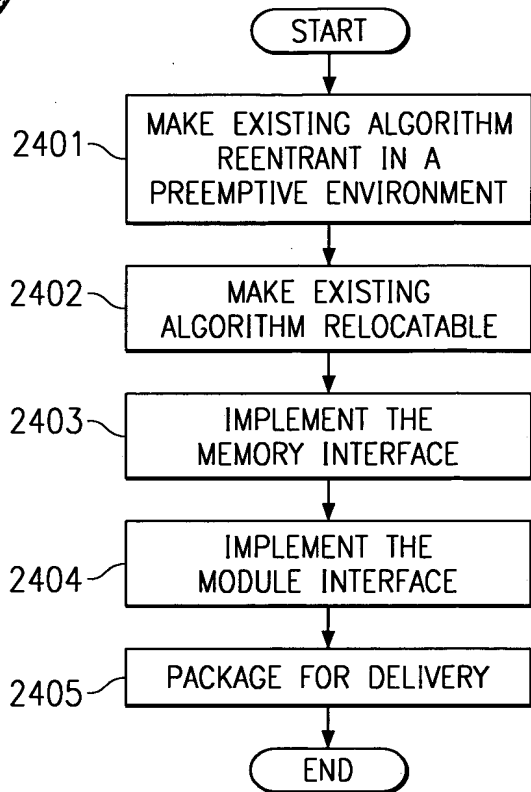


FIG. 26A

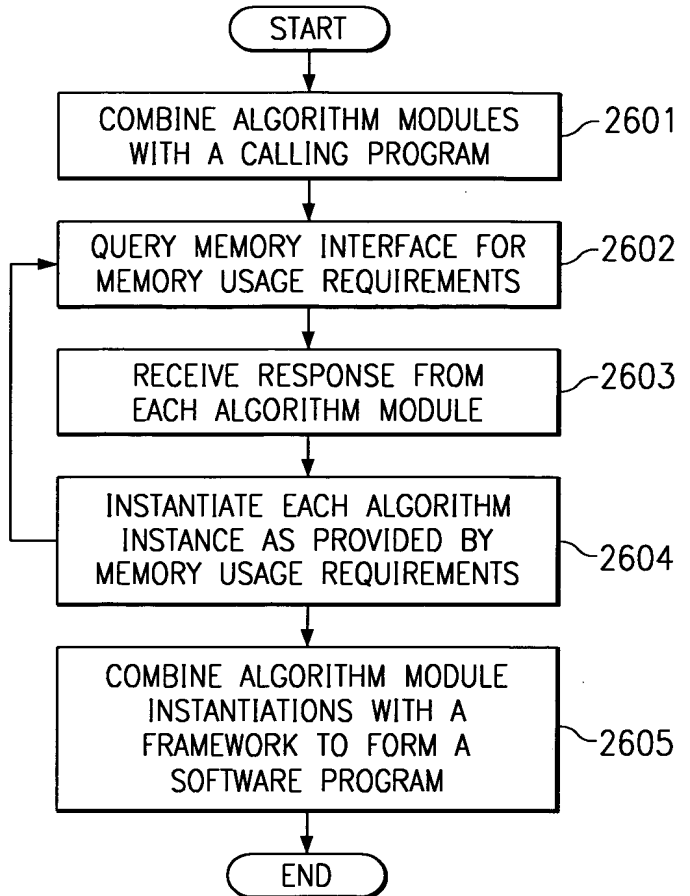


FIG. 25

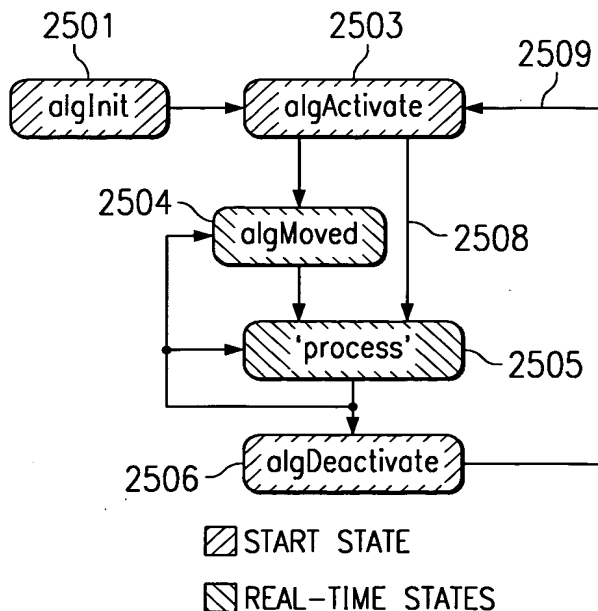
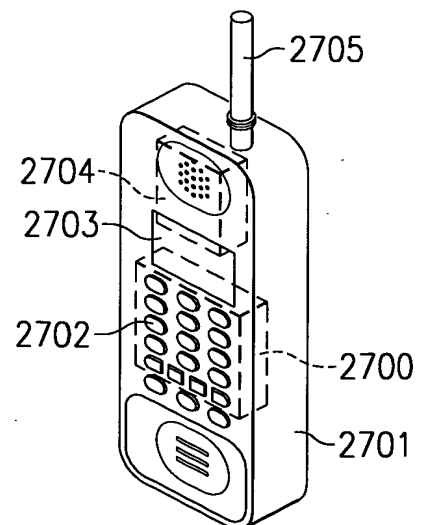


FIG. 27



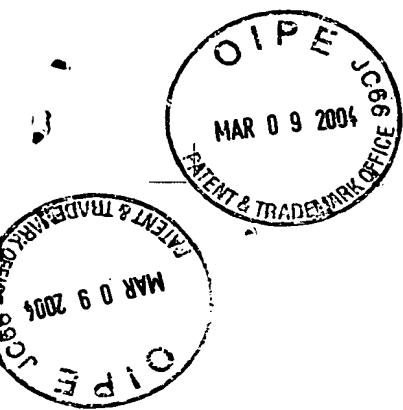
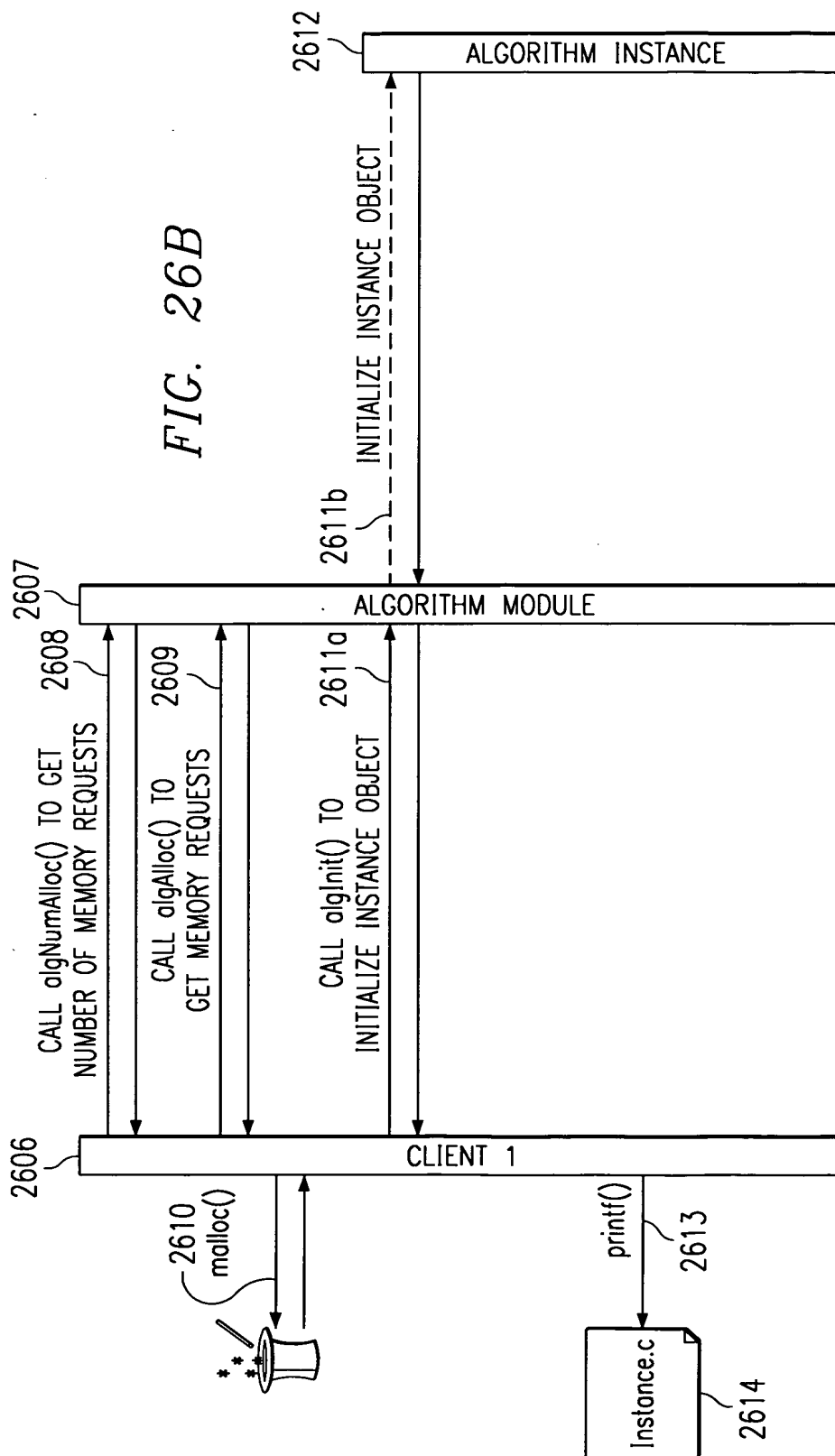


FIG. 26B



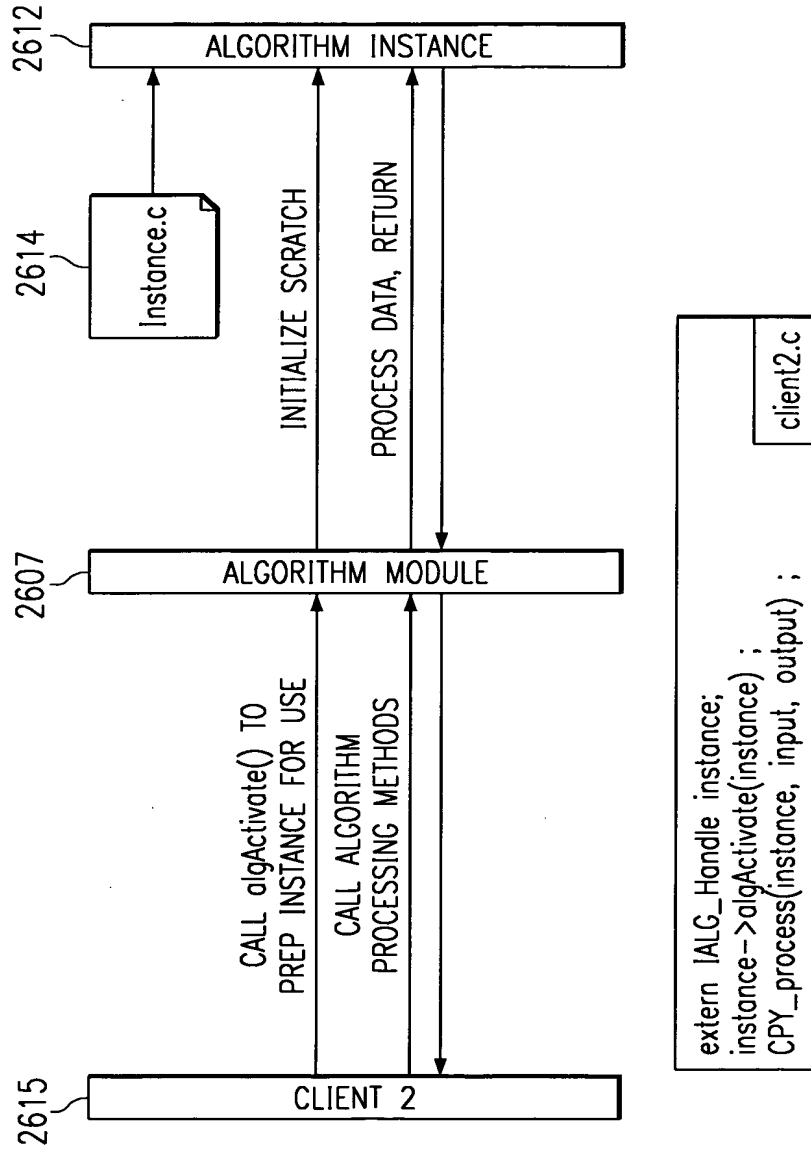


FIG. 26C



FIG. 1A

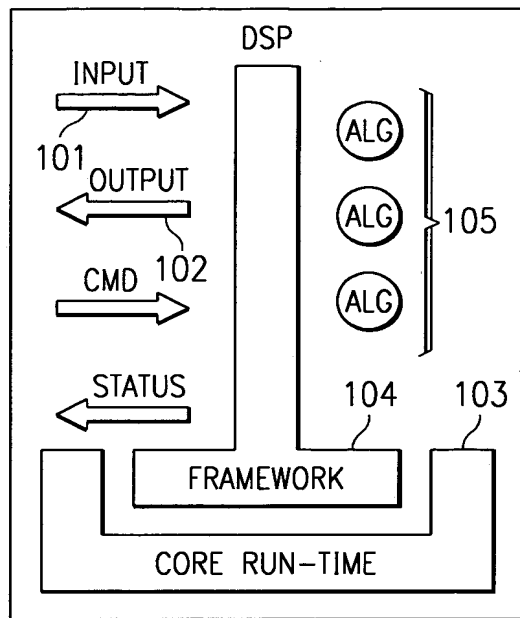


FIG. 1B

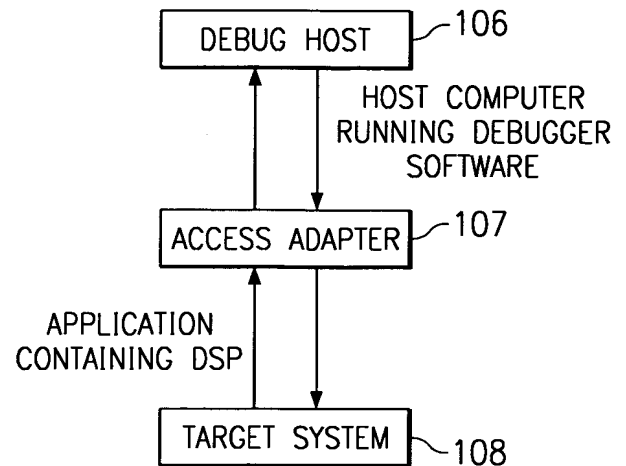
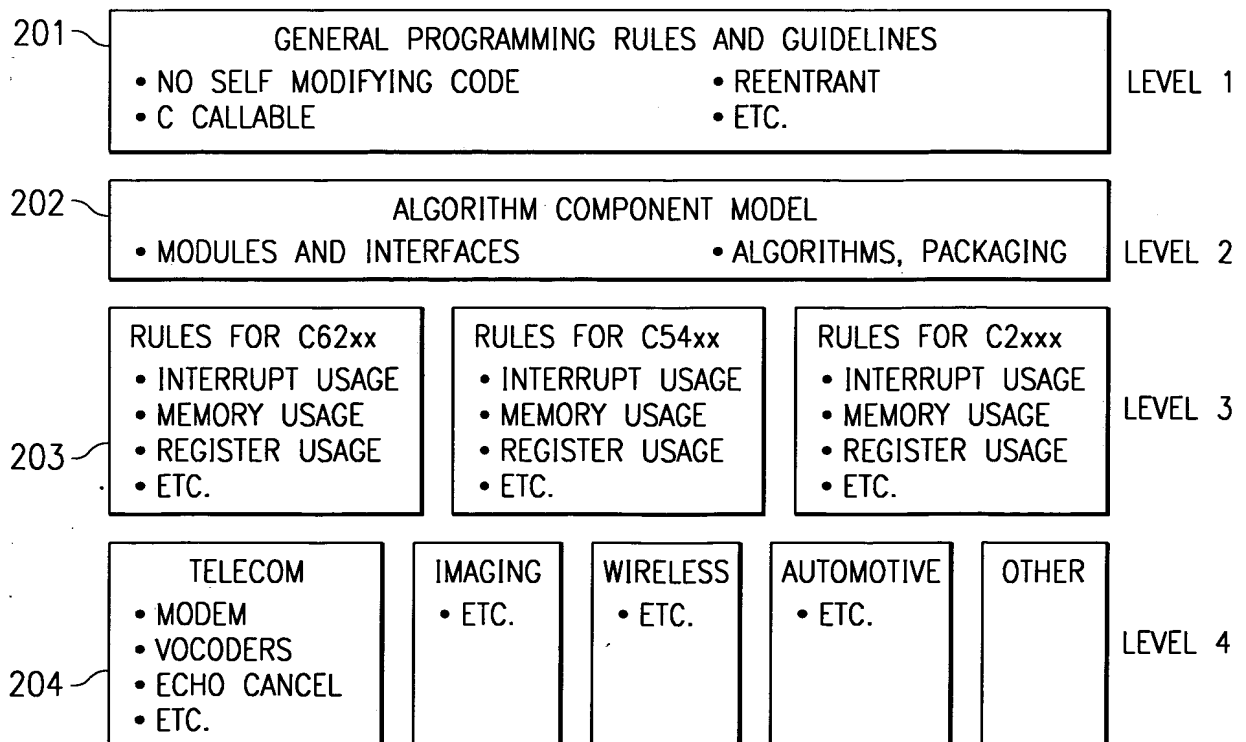


FIG. 2



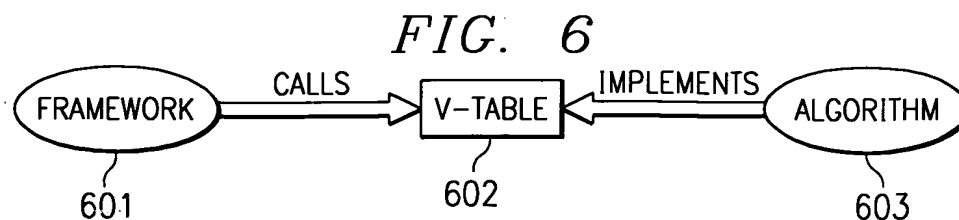
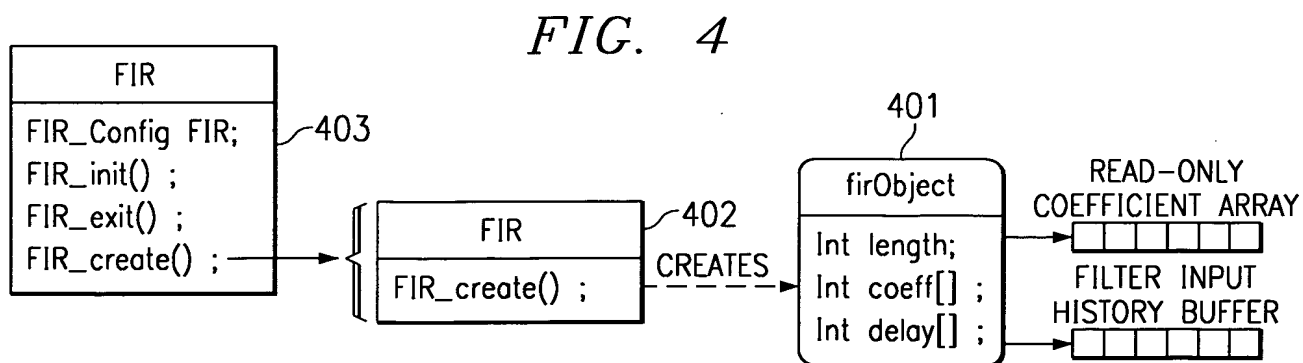
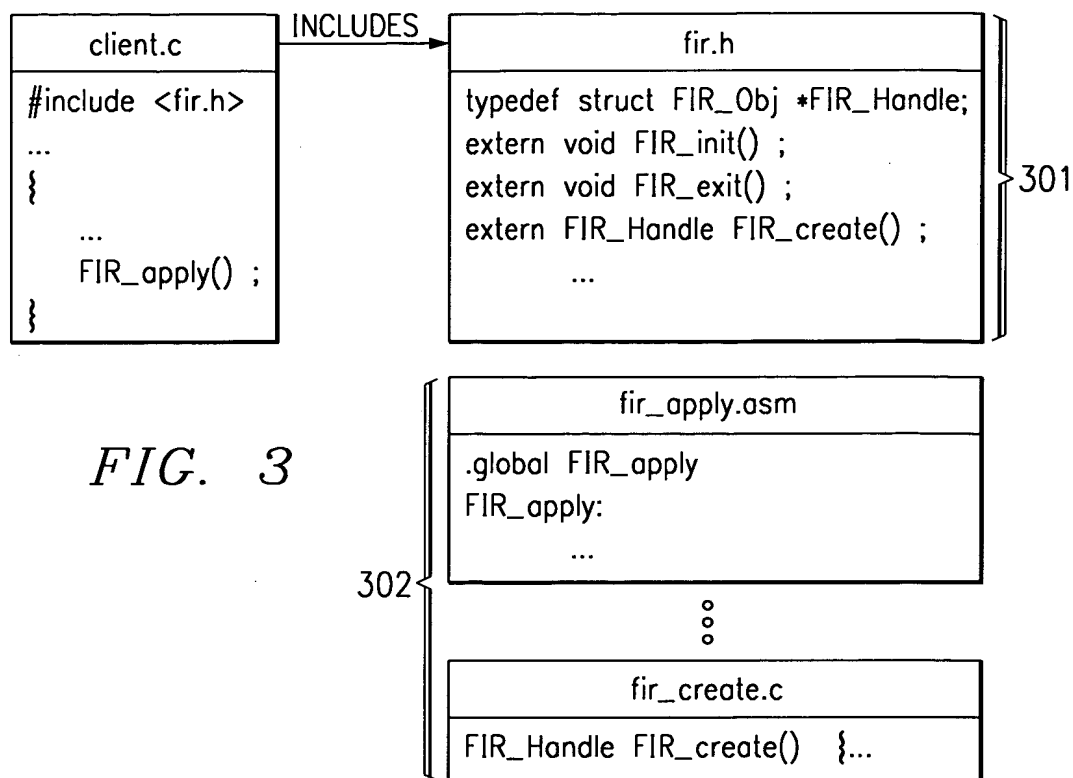




FIG. 5

```
501 { void FIR_init (void)
      {
      }
      void FIR_exit (void)
      {
      }

505 { typedef FIR_Params { /* FIR_Obj creation parameters */
      int frameLen ;      /* input/output frame length */
      int *coeff ;        /* pointer to filter coefficients */
      } FIR_Params ;

      FIR_Params FIR_PARAMS = { 64, NULL } ; /* default parameters */

      typedef struct FIR_Obj { /* FIR_Obj definition */
      int hist [16] ;        /* previous input value */
      int frameLen ;        /* input frame length */
      int *coeff ;
      } FIR_Obj ;

502 { FIR_Handle FIR_create (FIR_Obj *fir, const FIR_Params *params)
      {
      if (fir != NULL) {
      if (params == NULL) { /* use defaults if params is NULL */
      params = &FIR_PARAMS ;
      }
      fir->frameLen = params->frameLen ;
      fir->coeff = params->coeff ;
      memset(fir->hist, 0, sizeof (fir->hist)) ;
      }

      return (fir) ;
      }

503 { void FIR_delete (FIR_Handle fir)
      {
      }

504 { void FIR_apply (FIR_Handle fir, int in[] , int out[] )
      {
      int i ;

      /* filter data using coefficients fir->coeff and
      history fir->hist */
      for (i = 0 ; i < fir->frameLen; i++) {
      out [i] = filter(in[i], fir->coeff, fir->hist) ;
      }
      }
```



FIG. 7A

```

/*-----*/
/* TYPES AND CONSTANTS */
/*-----*/
706 ~ #define IALG_DEFMEMRECS 4
      #define IALG_OBJMEMREC 0
711 ~ #define IALG_SYSCMD 256
712 ~ #define IALG_EOK 0
713 ~ #define IALG_EFAIL -1
      typedef enum IALG_MemAttrs {
709 ~     IALG_SCRATCH,
        IALG_PERSIST,
        IALG_WRITEONCE
      } IALG_MemAttrs;

714 ~ #define IALG_MPROG 0x0008
      #define IALG_MXTRN 0x0010
/*
* ===== IALG_MemSpace =====
*/

708 ~ typedef enum IALG_MemSpace {
      IALG_EPROG = IALG_MPROG | IALG_MXTRN,
      IALG_IPROG = IALG_MPROG,
      IALG_ESDATA = IALG_MXTRN + 0,
      IALG_EXTERNAL = IALG_MXTRN + 1,
      IALG_DARAM0 = 0,
      IALG_DARAM1 = 1,
      IALG_SARAM = 2,
      IALG_SARAM0 = 2,
      IALG_SARAM1 = 3
    } IALG_MemSpace;

710 ~ typedef struct IALG_MemRec {
      Int size;
      Int alignment;
      IALG_MemSpace space;
      IALG_MemAttrs attrs;
      Void *base;
    } IALG_MemRec;

```

```

/* default number of memory records */
/* memory record index of instance object */
/* minimum "system" IALG_Cmd value */
/* successful return status code */
/* unspecified error return status code */

/* scratch memory */
/* persistent memory */
/* write-once persistent memory */

/* program memory space bit */
/* external memory space bit */

/* external program memory */
/* internal program memory */
/* off-chip data memory (accessed sequentially) */
/* off-chip data memory (accessed randomly) */
/* dual access on-chip data memory */
/* dual access on-chip data memory */
/* single access on-chip data memory */
/* block 0, equivalent to IALG_SARAM */
/* block 1, if independant blocks required */

/* 'sizeof' memory request in MAUs (minimum address-able unit) */
/* alignment requirement (in MAUs) */
/* allocation space */
/* memory attributes */
/* base address of allocated buf */

```

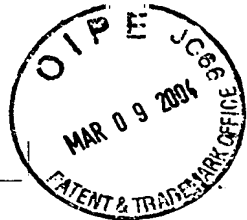


FIG. 7B

```
/*
 * ===== IALG_Obj =====
 * Algorithm instance object definition
 *
 * All XDAIS algorithm instance objects *must* have this structure as their first element.
 * However, they do not need to initialize it; initialization of this sub-structure is done by
 * the "framework".
 */

typedef struct IALG_Obj {
    struct IALG_Fxns *fxns; } 705
} IALG_Obj;

/*
 * ===== IALG_Handle =====
 * Handle to an algorithm instance object
 *
 */
typedef struct IALG_Obj *IALG_Handle; } 707
/*

 * ===== IALG_Params =====
 * Algorithm instance creation parameters
 * All XDAIS algorithm parameter structures *must* have a this as their first element.
 */

typedef struct IALG_Params {
    Int size; /* number of MAUs (i.e. the 'sizeof') the structure */ } 703
} IALG_Params;

/*
 * ===== IALG_Status =====
 * Pointer to algorithm specific status structure
 * All XDAIS algorithm status structures *must* have this as their first element.
 */

typedef struct IALG_Status {
    Int size; /* number of MAUs (i.e. the 'sizeof') the structure */ } 704
} IALG_Status;

/*
 * ===== IALG_Cmd =====
 * Algorithm specific command. This command is used in conjunction with IALG_Status to
 * get and set algorithm specific attributes via the algControl method.
 */
typedef unsigned int IALG_Cmd;
```


FIG. 7C

```

typedef struct IALG_Fxns {
    Void *implementationId; 702
    Void (*algActivate) (IALG_Handle);
    Int (*algAlloc) (const IALG_Params *, struct IALG_Fxns **, IALG_MemRec *);
    Int (*algControl) (IALG_Handle, IALG_Cmd, IALG_Status *);
    Void (*algDeactivate) (IALG_Handle);
    Int (*algFree) (IALG_Handle, IALG_MemRec *);
    Int (*algInit) (IALG_Handle, const IALG_MemRec *, IALG_Handle, const
    IALG_Params *);
    Void (*algMoved) (IALG_Handle, const IALG_MemRec *, IALG_Handle, const
    IALG_Params *);
    Int (*algNumAlloc) (Void);
} IALG_Fxns;
701

```

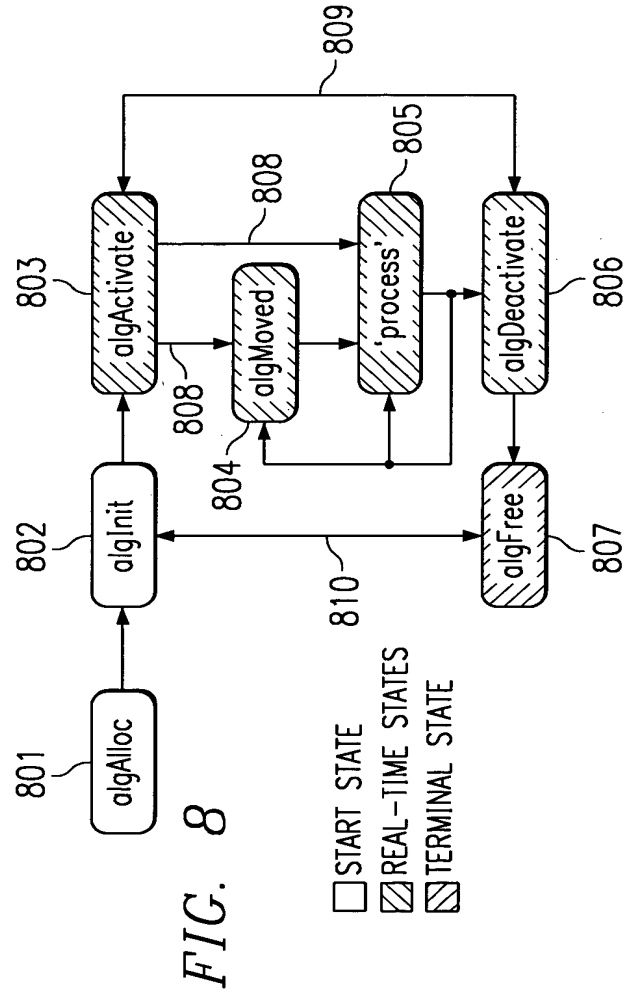




FIG. 9

```
client ()
{
    FIR_Params stdParams;
    FIR_TI_Params tiParams;

    stdParams = FIR_PARAMS;           /* initialize all fields to defaults */
    stdParams.coeff = ...;            /* initialize selected parameters */
    fxns->algAlloc(&stdParams, ...) ; /* pass parameters to algorithm */

    tiParams = FIR_TI_PARAMS;         /* initialize all fields to defaults */
    tiParams.coeff = ...;             /* initialize selected parameters */
    fxns->algAlloc(&tiParams, ...) ;  /* pass parameters to algorithm */
}

Int FIR_TI_algAlloc(IALG_Params *clientParams, ...)
{
    FIR_TI_Params params = FIR_TI_PARAMS;

    /* client passes in parameters, use them to override defaults */
    if (clientParams != NULL) {
        memcpy(&params, clientParams, clientParams->size) ;
    }

    /* use params as the complete set of parameters */
    :
}
}
```

FIG. 11

```
Void FIR_apply (FIR_Handle alg, Int *in[], Int *out[] )
{
    /* do app specific initialization of scratch memory */
    if (alg->fxns->ialg.algActivate != NULL) {
        alg->fxns->ialg.algActivate(alg) ;
    }
    /* filter data */
    alg->fxns->filter(alg, in, out) ;

    /* do app specific store of persistent data */
    if (alg->fxns->ialg.algDeactivate != NULL) {
        alg->fxns->ialg.algDeactivate(alg) ;
    }
}
}
```



FIG. 10

```
#define MAXMEMRECS 16

typedef struct ALG_Obj {
    IALG_Fxns  fxns;          /* algorithm functions */
} ALG_Obj;

IALG_Handle ALG_create (IALG_Fxns *fxns, IALG_Params *params)
{
    IALG_MemRec  memTab [MAXMEMRECS] ;
    IALG_Handle  alg = NULL;
    Int          n;

    if (fxns->algNumAlloc () <= MAXMEMRECS) {
        n = fxns->algAlloc(params, memTab) ;
        if (allocMemory(memTab, n)) {

            alg = (IALG_Handle)memTab[0] .base;
            alg->fxns = fxns ;
            if (fxns->algInit(alg, memTab, params) != IALG_EOK) {
                fxns->algFree(alg, memTab) ;
                freeMemory(memTab, n) ;
                alg = NULL;
            }
        }
    }

    return (alg) ;
}

Void ALG_delete (IALG_Handle alg)
{
    IALG_MemRec memTab [MAXMEMRECS] ;
    Int n;

    n = alg->fxns->algFree(alg, memTab) ;
    freeMemory(memTab, n) ;
}
```

1001 {

1002 {